Apuntes

Simulación de Coordenadas 3D

Para simular la coordenada z a partir de (x, y) usando métodos de kernel y el Teorema de Representación de Riesz, primero se asume que z es una función f(x, y) definida en un espacio de Hilbert con núcleo reproducible (RKHS) asociado a un kernel K((x, y), (u, v)) como el RBF o polinómico; el Teorema de Riesz garantiza que cualquier funcional lineal y continuo puede representarse como un producto interno con un único vector, y el Representer Theorem (basado en Riesz) establece que la solución óptima f al problema de ajustar z en el RKHS siempre es una combinación lineal de K((x_i, y_i), (x, y)) para los puntos de entrenamiento i; así, si se tienen n puntos 2D con valores de z (o se imponen restricciones que determinen z en ciertos puntos), se construye la matriz de Gram K de tamaño n por n con K_ij = K((x_i, y_i), (x_j, y_j)), se resuelve un problema de regresión regularizada (por ejemplo Kernel Ridge o SVR) para encontrar los coeficientes alfa que definen f, y finalmente, para cualquier (x, y) nuevo, z se obtiene como suma de alfa_i por K((x_i, y_i), (x, y)), lo cual permite "inventar" o aproximar la coordenada z sin proyectar explícitamente los datos en un espacio de grandes dimensiones sino usando el kernel de forma implícita.

Inferencias sobre señales continuas

A la hora de detectar y reconocer gestos (por ejemplo, en lengua de señas) dentro de una corriente continua de datos sensoriales (como guantes con sensores o una cámara con keypoints), es clave tener una estrategia de segmentación. A continuación se describen varias propuestas, incluyendo la que tú planteas de ventanas deslizantes (sliding window) y otras aproximaciones que se usan comúnmente en señales continuas.

1. Ventana móvil (sliding window)

a. Dividir la señal en chunks de tamaño fijo (por ejemplo, 2 segundos, 1 segundo, etc., dependiendo del framerate).

b. Desplazar esta ventana un paso pequeño (por ejemplo, 100 ms, 200 ms, etc.) y en cada posición se recolectan todos los datos (frames/samples) que caen en la ventana.

c. Inferir con un modelo de clasificación (por ejemplo, STGCN u otro) si en esa ventana se está realizando un gesto específico.

d. Filtrar salidas de baja confianza (por ejemplo, descartar predicciones con confianza < umbral).

e. Unir/Integrar predicciones cercanas en el tiempo para producir un único gesto detectado.

Pros y contras

- Pros:
    - Es sencillo de implementar.
    - El modelo no necesita saber explícitamente dónde empieza/termina un gesto; solo recibe ventanas de datos.
    - Permite un funcionamiento “online” relativamente directo: cada ventana se procesa apenas está completa.
- Contras:
    - La segmentación no es explícita; podría recortar gestos en medio de una ventana o generar duplicación de gestos si se superponen en varias ventanas.
    - Latencia: cada ventana añade un retardo (el tamaño de la ventana).
    - Se debe diseñar un método de post-procesado para unificar predicciones consecutivas (p. ej., un suavizado o un “votado” en el tiempo).

Ejemplo de flujo

1. Tienes un buffer de frames/samples de la mano (cada frame con 21 keypoints x 3D, o datos de los sensores del guante).
2. Cada vez que el buffer alcanza window_size frames, se corre una inferencia.
3. Desplazas la ventana hop_size frames y vuelves a inferir.
4. Guardas la confianza y la clase predicha.

5. Si la confianza supera cierto umbral (p. ej. 0.8), se “activa” el gesto.
6. Para evitar que la ventana siguiente vuelva a disparar el mismo gesto, puedes usar:
    - un “cool-down” (p. ej., no aceptar otro gesto idéntico hasta pasados N frames),
    - o un suavizado temporal: solo si se sostiene el gesto por varias ventanas, lo confirmas.

2. Modelo de detección de “estado idle” vs. “gesto”

Una variante que evita “forzar” un tamaño de ventana fijo es entrenar o diseñar un sistema que:

a. Tenga dos etapas o un modelo multicategoría adicional que incluya la clase “sin gesto” (idle).
b. Cuando el modelo detecta que se pasó de “idle” a “gesto”, empieza a grabar frames.
c. Una vez vuelve a “idle”, se asume que el gesto terminó, y ese chunk se manda a un clasificador final.

Pros y contras

- Pros:
    - Reduces el overhead de correr clasificación en cada ventana. Solo se dispara la clasificación cuando se ha detectado que se está en un “estado no idle”.
    - Es más natural segmentar: “idle → gesto → idle”.
- Contras:
    - Requiere un modelo fiable que discrimine entre “idle” y “gesto”.
    - Si el guante se “mueve” (ruido, o no se saca la mano) y no pasa por idle, puede no resetearse la captura.
    -

3. Modelos de “detección de gestos” en streaming

Existen arquitecturas diseñadas específicamente para la detección online de acciones o gestos en secuencias de vídeo o señales. Por ejemplo:

a. Temporal Action Detection (del mundo de la visión por computador), donde el modelo no solo clasifica sino que localiza en el tiempo (start frame, end frame).

b. Temporal Convolutional Networks (TCN) o transformers streaming que, a medida que reciben frames, emiten un puntaje de qué gesto se está realizando y cuándo termina.

c. CTC-based approaches (Connectionist Temporal Classification) adaptados, donde se modela secuencia a secuencia y se saca un alineamiento temporal.

El problema aquí es más complejo, pero evita la necesidad de ventanas fijas, pues el modelo aprende a detectar "fronteras" de inicio/fin.

4. Ventana móvil + detección refinada

Una estrategia intermedia (muy usada en "online recognition"):

a. Mantener un buffer de tamaño mayor que un gesto típico.

b. A medida que fluyen datos, vas actualizando el buffer.

c. Haces inferencia cada cierto número de frames (p. ej., cada 5 frames).

d. Al detectar un aumento fuerte en la confianza de cierta clase, "centras" la ventana alrededor de ese pico para refinar la clasificación (por ejemplo, usando un chunk ligeramente más grande) y decides si disparar la clase o no.

Esta combinación te da cierta flexibilidad y un post-procesado más sofisticado para reducir falsos positivos.

5. Consideraciones

a. Tamaño de ventana:

   - Depende de la duración típica del gesto.
   - Si un gesto dura ~2 segundos, y capturas a 30 fps, estás hablando de ~60 frames. Quizás usas ventanas de 60 frames con un salto de 10 frames.

b. Smoothing (suavizado) de predicciones:
    - En tiempo real, es muy útil llevar un historial de predicciones de las últimas N ventanas y decidir una salida final de forma consensuada (por ejemplo, majority voting en un sliding window de predicciones).
    - Evita que ruidos puntuales disparen una clase equivocada.
c. Umbral de confianza:
    - Selecciona un umbral alto (0.7–0.9) si prefieres precisión (menos falsos positivos).
    - Selecciona un umbral más bajo si prefieres sensibilidad (detectar gestos a costa de más falsos positivos).
d. Clase "nula":
    - Entrena tu modelo con una clase "ningún gesto" para que sea capaz de descartar ventanas que no contengan un gesto válido.
e. Etiquetado de datos continuo:
    - Necesitarás (en entrenamientos más sofisticados) secuencias largas etiquetadas con la localización temporal de cada gesto.
    - Una forma rápida (aunque menos precisa) es capturar un video/sesión de guante, segmentar a mano el inicio-fin de cada gesto y etiquetarlo.

REFERENCIAS

1. **Segmentación y Reconocimiento de Gestos con Ventanas Deslizantes**
    - **Starner, T., & Pentland, A. (1995). "Real-time American Sign Language recognition from video using hidden Markov models."** *MIT Media Laboratory Perceptual Computing Section Technical Report*.
        - Pioneros en usar métodos de reconocimiento continuo. El acercamiento de HMM incluía "ventanas" sucesivas de frames para la detección y clasificación de gestos.
2. **Detección Temporal de Acciones** (dónde empieza y termina el gesto):

- **Lea, C., Flynn, M. D., Vidal, R., Reiter, A., & Hager, G. D. (2017). "Temporal convolutional networks for action segmentation and detection." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 156–165).**
    - Proponen redes convolucionales temporales (TCN) para segmentar y localizar acciones en secuencias de video. El enfoque puede adaptarse al reconocimiento de gestos en lenguaje de señas.
- **Farha, Y. A., & Gall, J. (2019). "MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3575–3584).**
    - Extiende la idea con múltiples etapas de TCN para refinar la segmentación de acciones a lo largo del tiempo.

3. **Modelos "CTC-based"** (detección de secuencias sin segmentación explícita):
    - **Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). "Connectionist temporal classification: Labeling unsegmented sequence data with recurrent neural networks." In *Proceedings of the 23rd International Conference on Machine Learning (ICML)* (pp. 369–376).**
        - Presentan la metodología CTC, que se ha aplicado en reconocimiento de voz y puede también usarse para segmentar y reconocer gestos en secuencias continuas sin anotar inicios y finales.
4. **Recopilaciones/Síntesis en Sign Language Recognition (SLR)**:
    - **Cooper, H., Holt, B., & Bowden, R. (2011). "Sign language recognition." In *Visual Analysis of Humans* (pp. 539–562). Springer, London.**
        - Ofrece una revisión de técnicas clásicas para SLR (modelos ocultos de Markov, segmentación manual vs. automática, etc.).
    - **Koller, O. (2015). "Quantitative survey of the state of the art in sign language recognition." *arXiv preprint arXiv:1512.06434*.**

        - Analiza metodologías y conjuntos de datos disponibles en SLR, incluyendo segmentación y reconocimiento continuo.

5. **Sliding Window e Integración con Post-procesado**
    - Aunque no existe un solo paper que defina por completo el método de "ventana deslizante + post-procesado con umbrales de confianza", es un **patrón muy frecuente** en tareas de reconocimiento de acciones en streaming.
    - Por ejemplo, en sistemas de gesture recognition basados en sensores inerciales se usa ventaneo fijo:
        - **Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., & Marca, D. (2006). "Accelerometer-based gesture control for a design environment." Personal and Ubiquitous Computing, 10(5), 285–299.**